Light Logics and Optimal Reduction: Completeness and Complexity

Patrick Baillot*
LIPN, CNRS & Université Paris Nord
pb@lipn.univ-paris13.fr

Paolo Coppola†
Università di Udine
coppola@uniud.it

Ugo Dal Lago‡
Università di Bologna
dallago@cs.unibo.it

Abstract

Typing of lambda-terms in Elementary and Light Affine Logic (EAL , LAL resp.) has been studied for two different reasons: on the one hand the evaluation of typed terms using LAL (EAL resp.) proof-nets admits a guaranteed polynomial (elementary, resp.) bound; on the other hand these terms can also be evaluated by optimal reduction using the abstract version of Lamping's algorithm. The first reduction is global while the second one is local and asynchronous. We prove that for LAL (EAL resp.) typed terms, Lamping's abstract algorithm also admits a polynomial (elementary, resp.) bound. We also show its soundness and completeness (for EAL and LAL with type fixpoints), by using a simple geometry of interaction model (context semantics).

1 Introduction

Background and Motivations. Light logics such as Light Affine Logic (LAL) and Elementary Affine Logic (EAL) have been introduced in [18, 4] and then studied as type systems for lambda-calculus [3, 11, 12, 6, 9, 10] and semantically (*e.g.* in [24, 27, 13]). Their analysis has been motivated by two distinct features:

1. *Complexity-certified reduction*: using the syntax of *proof-nets*, the logics LAL (EAL, respectively) ensure that the program terminates with a polynomial (elementary, respectively) time bound;
2. *Simple optimal reduction*: for lambda-terms typed in these systems one can use the abstract version of Lamping's algorithm [23], without the so-called *oracle*, so plain graph rewriting with local rules.

However each of these approaches has its drawbacks:

- Proof-net reduction is global, requires the handling of *boxes*, and thereby enforces a form of synchronicity which is awkward to implement;
- Optimal reduction is local, asynchronous – and elegant, but . . . does not offer any guaranteed complexity bound.

Actually, even the fact that the abstract Lamping algorithm is *correct* for terms of EAL and LAL is not completely straightforward: it was pointed out since [4] and is part of the folklore, but it seems no explicit direct proof of this statement is available in the literature, although the general techniques of [21] could be applied to this restricted setting.

The goal of this paper is therefore to bring together the best of these two worlds: we prove that terms typable in LAL (EAL, respectively) can be reduced by the Lamping abstract algorithm with a certified polynomial (elementary, respectively) time bound. Moreover a type derivation in LAL or EAL carries two kinds of information: the sharing information and the boxing information. We actually show here that the boxing information is *not* needed to perform Lamping's abstract algorithm. Some systems like DLAL [9] or restrictions of EAL [11] do not use sharing: in that case knowing that the term is typeable (without knowing the type) is sufficient to apply the abstract algorithm.

Actually the bounds of light logics can also be obtained without the proof-net machinery, in plain lambda-calculus, if one considers fragments of the type systems, possibly with restricted (lazy) reduction [9, 10]. However this is still a global

*Partially supported by projects NO-CoST (ANR, JC05_43380), CRISS (ACI).
†Partially supported by a visiting professorship of Univ. Paris Nord.
‡Work partially carried out while the author was postdoctoral fellow of Univ. Paris Nord and of Univ. Denis Diderot, Paris 7.

form of reduction (β-reduction). Here we aim to handle the full type systems and to switch to a local reduction, which is motivating for concrete implementations and in particular for distributed evaluation [28].

Optimal Reduction and Light Logics. The fact that EAL typable terms can be reduced with Lamping's abstract algorithm is quite remarkable, since it is known that the bookkeeping needed for the oracle causes inefficiencies in optimal reduction [25].

On the other hand, proof-net reduction in these systems is performed with guaranteed complexity bound, one might think that the preservation of bounds when switching from proof-net reduction to optimal reduction is a consequence of optimality itself. However this is actually not true: the optimality concerns the number of parallel beta-steps, which is not directly related to the normalisation time [2, 1]. For an in-depth study of optimal reduction one can consult [5].

Moreover, techniques used when analyzing proof-net (or lambda-term) reduction time cannot be directly applied here. In particular, the level-by-level reduction strategy (see [18, 3]) has no counterpart in the framework of sharing graphs, where copying is done incrementally.

Contributions. Our main results are:

- We define a general class of admissible translations from light logics type derivations to sharing graphs, subsuming the ones proposed before.
- For each admissible translation, we show that graph reduction is sound and complete with respect to lambda-reduction.
- Moreover, we show that graph reduction can be performed in bounded time, where the bound is of the same complexity order as the one we have on the underlying logical system.

Moreover we believe that the main technique used to prove the complexity bounds (Section 6), based on the definition of *weights* for sharing graphs (or interaction nets, [22]) following [14] is of its own interest and could presumably be applied to other problems.

2 Soundness and Completeness in the General Case

Before introducing the specific logical systems we are interested in, we define the notions of soundness and completeness for abstract systems of graph reduction. Throughout the paper, Λ is the set of pure, untyped, lambda terms. If A is a set and $\rightarrow$ is a binary relation on A, the set of normal forms in A (with respect to $\rightarrow$) will be denoted $NF(A)_{\rightarrow}$.

Definition 1 (Graph Rewriting Systems) *A Θ-graph rewriting system is a quintuple $(\Theta, \Delta, \rightarrow_{\Delta}, \mathcal{T}, \mathcal{R})$ where:*

- *$\Theta \subseteq \Lambda$ is a set of lambda-terms to which the technique can be applied.*
- *Δ is a set of* graphs*.*
- *$\rightarrow_{\Delta}$ is a rewriting relation on Δ.*
- *$\mathcal{T}$ is a total binary relation from Θ to Δ, called the* initial translation*.*
- *$\mathcal{R}$ is a function from Δ to Λ, called the* readback*.*

Note that $\mathcal{T}$ is a relation and not a mere function, since we want to allow several possible translations of a term (this is related to the fact that we will allow the possibility to decorate a given lambda-term as several different proof-nets).

Definition 2 (Soundness) *We say that the Θ-graph rewriting system $(\Theta, \Delta, \rightarrow_{\Delta}, \mathcal{T}, \mathcal{R})$ is* sound *with respect to a reduction relation $\rightarrow$ on Λ iff for every term $t \in \Theta$, if $G \in \mathcal{T}(t)$ and G reduces to normal form H (in $\rightarrow_{\Delta}$) then t reduces to normal form u (in $\rightarrow$) and $\mathcal{R}(H) = u$:*

$$\begin{tikzcd} t \arrow[r, dashed, "*"] \arrow[d, "\mathcal{T}"] & u \\ G \arrow[r, "*", "\Delta"'] & H \arrow[u, "\mathcal{R}"'] \end{tikzcd}$$

Soundness of a Θ-graph rewriting system implies that if we start with a term t in Θ, translate it into a graph, reduce the graph and finally read-back a term u, then u is the normal form of t. This does not mean the Θ-graph rewriting system will necessarily do its job: to be sure about that, we need completeness:

Axiom, Cut and Structural Rules

$$\frac{}{x : A \vdash x : A}\ A \qquad \frac{\Gamma \vdash t : A \quad \Delta, x : A \vdash u : B}{\Gamma, \Delta \vdash u\{t/x\} : B}\ U$$

$$\frac{\Gamma \vdash t : B}{\Gamma, x : A \vdash t : B}\ W \qquad \frac{\Gamma, x :!A, y :!A \vdash t : B}{\Gamma, z :!A \vdash t\{z/x, z/y\} : B}\ X$$

Multiplicative Logical Rules

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\ R_{\multimap} \qquad \frac{\Gamma \vdash t : A \quad \Delta, x : B \vdash u : C}{\Gamma, \Delta, y : A \multimap B \vdash u\{yt/x\} : C}\ L_{\multimap}$$

Exponential Rules

$$\frac{\Gamma \vdash t : A}{!\Gamma \vdash t :!A}\ P_!$$

Second Order Rules

$$\frac{\Gamma \vdash t : A \quad \alpha \notin FV(\Gamma)}{\Gamma \vdash t : \forall\alpha.A}\ R_\forall \qquad \frac{\Gamma, x : A\{B/\alpha\} \vdash t : C}{\Gamma, x : \forall\alpha.A \vdash t : C}\ L_\forall$$

Least Fixpoint Rules

$$\frac{\Gamma \vdash t : A\{\mu\alpha.A/\alpha\}}{\Gamma \vdash t : \mu\alpha.A}\ R_\mu \qquad \frac{\Gamma, x : A\{\mu\alpha.A/\alpha\} \vdash t : B}{\Gamma, x : \mu\alpha.A \vdash t : B}\ L_\mu$$

Figure 1. A sequent calculus for elementary linear logic with second order and fixpoints.

Definition 3 (Completeness) *We say that the Θ-graph rewriting system $(\Theta, \Delta, \to_\Delta, \mathcal{T}, \mathcal{R})$ is* complete *with respect to a reduction relation $\to$ on Λ iff for every term $t \in \Theta$ if t reduces to normal form u, then any $G \in \mathcal{T}(t)$ reduces to normal form H, where $\mathcal{R}(H) = u$.*

$$\begin{tikzcd} t \arrow[r, "*"] \arrow[d, "\mathcal{T}"] & u \\ G \arrow[r, dashed, "*", "\Delta"'] & H \arrow[u, "\mathcal{R}"'] \end{tikzcd}$$

3 Type Assignment Systems and Proof-Nets

Formulae of (intuitionistic) *elementary affine logic* (EAL for short) are generated by the following productions:

$$A ::= \alpha \mid A \multimap A \mid !A \mid \forall\alpha.A \mid \mu\alpha.A$$

where α ranges over a countable set of *atoms*. Recall that ! is called an *exponential connective* or *modality*.

Here we are considering in fact EAL with type fixpoints (recursive types) but this does not modify its normalisation properties [15]. Most references in the literature deal with the second order fragment $\mathsf{EAL}_\forall$, which does not include type fixpoints.

EAL can be seen as a type system for terms in Λ: Figure 1 presents type assignment by means of sequent calculus, which is tedious for typing but convenient for studying the dynamics. Other presentations of typing can be found in the literature [11, 12, 7]. Note that sharing is allowed, for instance by using rules X and U. Θ_{EAL} denotes the set of lambda terms which are typable in elementary affine logic.

Elementary affine logic proofs can be formulated as a system of (intuitionistic) proof-nets Δ_{EAL}, defined inductively on Figure 2. Node X is called a *contraction* node. The *principal edge* of a node v is the edge incident to v through its *principal port* (indicated with a $\bullet$). A *cut* is an edge $e = \{v, w\}$ which is principal for both v and w.

EAL proof-nets can be endowed with a rewriting relation $\to_{\mathsf{EAL}}$ (see Figure 3). The important case of $\to_{\mathsf{EAL}}$ is when an X node meets a $R_!$ node, corresponding to a box: in this case the box is duplicated and the doors $L_!$ of the two copies are linked to X nodes (*contraction normalisation step*).

If v (resp. e) is a node (resp. edge) of a proof-net, $\partial(v)$ (resp. $\partial(e)$) denotes its *box-depth* (*level*). If $G \in \Delta_{\mathsf{EAL}}$ is a proof-net, its depth $\partial(G)$ is the maximal depth of its edges. The *stratification property* of EAL states that the depth $\partial(e)$ of an edge does not change through $\to_{\mathsf{EAL}}$.

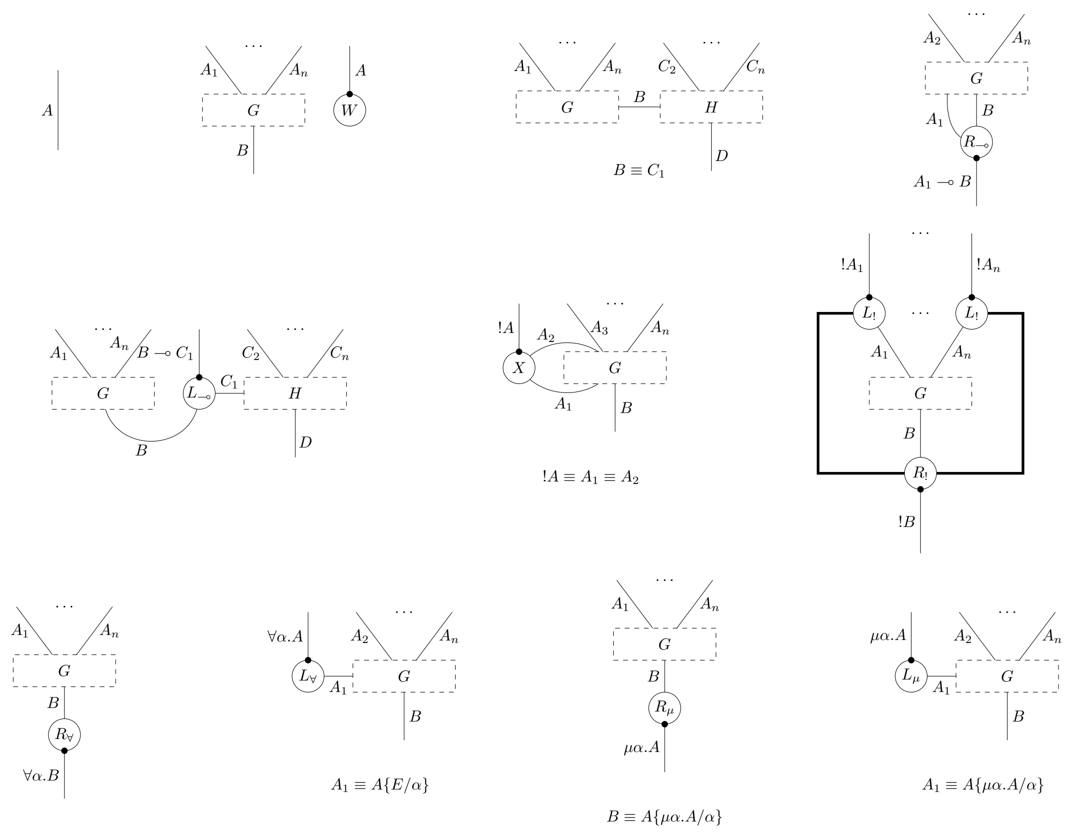


Figure 2. Proof-nets for elementary affine logic.

Light Affine Logic. LAL can be obtained from EAL by adopting a stricter exponential discipline: one restricts the rule $P_!$ of EAL to the case where Γ contains at most one formula, but also adds a new connective $\S$ with rule $P_\S$ (see Figure 4). The connective $\S$ is a weak form of !, that does not allow for contraction (rule X).

There is a translation $(.)^e$ from LAL to EAL formulae obtained by replacing $\S$ with !. It extends to a translation on proofs.Therefore the set Θ_{LAL} of lambda-terms typable in LAL is included in Θ_{EAL}.

The proofs-nets of LAL are defined as those of EAL but with two new nodes $L_\S$ and $R_\S$ and conditions on boxes: a box with $R_!$ main door (!-box) can have at most one $L_!$ door; a box with $R_\S$ main door ($\S$-box) can have any number of $L_\S$ and $L_!$ doors. A rewriting relation $\rightarrow_{\mathsf{LAL}}$ is defined on these proof-nets [14]. This reduction does not cause any duplication of a $\S$-box.

The translation $(.)^e$ can be extended naturally to a translation from LAL to EAL proof-nets, and it maps $\rightarrow_{\mathsf{LAL}}$ to $\rightarrow_{\mathsf{EAL}}$. Therefore the set of LAL proof-nets can be seen as a subset of Δ_{EAL}. Hence properties of EAL proof-nets will be valid in particular for LAL proof-nets and we will state them only for EAL (except for complexity issues in Section 6).

Paths. A *direct path* is a sequence of edges $e_1, \ldots, e_n$ such that the following conditions hold:

- For every $1 \leq i < n$, e_i and e_{i+1} have a vertex v_i in common;
- For every $1 \leq i < n$, $e_i \neq e_{i+1}$ and either e_i or e_{i+1} is principal for v_i.

An example of a direct path is reported in Figure 5(a). We say that a direct path $e_1, \ldots, e_n$ with $n \geq 2$ *starts* at v iff $e_1 = \{v, w\}$ is principal for v and there is z with $e_2 = \{w, z\}$. A direct path $e_1, \ldots, e_n$ is *simple* iff for every $1 \leq i < n$, the edge e_{i+1} is principal for v_i. The direct path in Figure 5(b) is simple, while the one in Figure 5(a) is not. A direct path is *maximal* iff it is not part of any longer direct path. Two edges e, g are *non-consecutive* iff there cannot be any direct path in the form e, g (or, equivalently, in the form g, e). A box b in a proof-net N is *special* iff any direct path starting from one of its premises is simple.

Lemma 1 *Any non-simple direct path $e_1, \ldots, e_n$ starting at any node v contains a cut e_i such that $\partial(e_i) \leq \partial(e_1)$.*

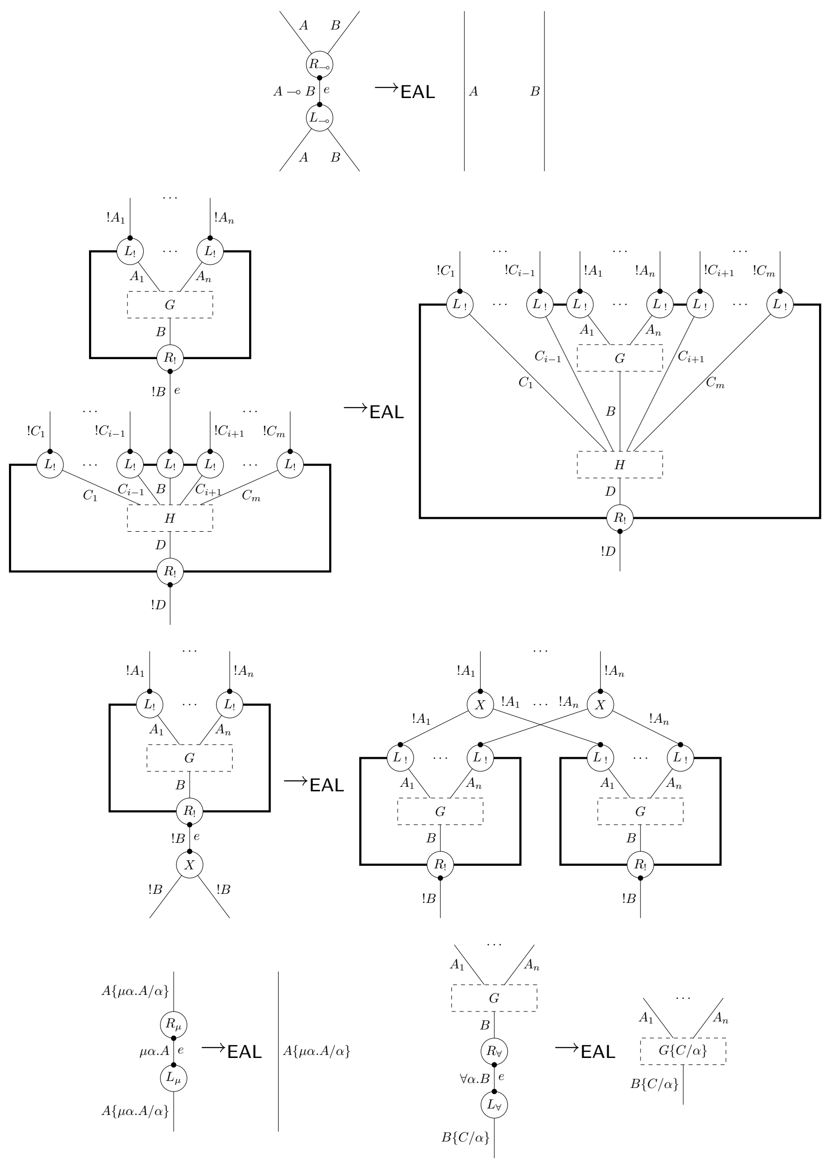


Figure 3. Rewriting rules for elementary affine logic proof-nets.

$$\frac{\vdash t : A}{\vdash t : !A}\ P^1_! \qquad \frac{x : A \vdash t : A}{x : !A \vdash t : !A}\ P^2_! \qquad \frac{\Gamma, \Delta \vdash t : A}{\S\Gamma, !\Delta \vdash t : \S A}\ P_\S$$

Figure 4. Exponential rules of light affine logic with second order and fixpoints.

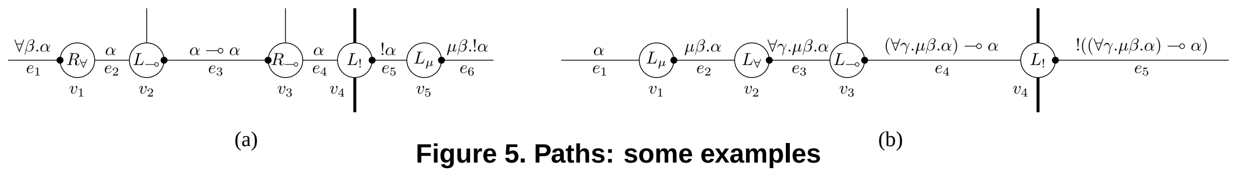


Figure 5. Paths: some examples

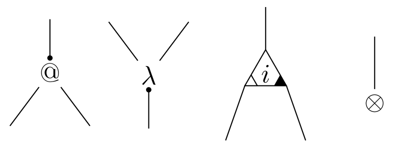


Figure 6. Sharing Nodes.

Proof. As a preliminary fact, take notice that for any simple direct path $e_1, \ldots, e_n$, it holds that $\partial(e_n) \leq \partial(e_1)$. Indeed, you can enter a box only through a principal port. We can prove the lemma by induction on n:

- If $n = 1$, then the path is simple.
- Then, observe that any non-simple, direct path e_1, e_2 starting in v contains a cut, namely e_1. Indeed, by definition e_1 is principal for v and, since the path is non-simple, e_1 is principal for v_1, too.
- Let $n \geq 3$ and $e_1, \ldots, e_n$ be a non-simple direct path starting from v. If $e_1, \ldots, e_{n-1}$ is non-simple, then by inductive hypothesis, it contains a cut. If $e_1, \ldots, e_{n-1}$ is simple and $e_1, \ldots, e_n$ is not simple, then e_{n-1} is principal for v_{n-2} and v_{n-1}. As a consequence, e_{n-1} is a cut. Moreover, $\partial(e_{n-1}) \leq \partial(e_1)$.

This concludes the proof. □

Strategies. There are two reduction strategies for proof-nets in Δ_{EAL} (or Δ_{LAL}) that are of particular interest for our purposes:

- The *level-by-level strategy*, LBL. A cut at level $n + 1$ cannot be reduced if there are cuts at level n.
- The *modified level-by-level strategy*, MLBL. It is the level-by-level one with an additional constraint: whenever we copy a box b, b must be a special box.

Notice that MLBL is a reduction strategy due to Lemma 1. Indeed, if a box b is involved in a cut e but is not special, then we can find another cut g such that $\partial(g) \leq \partial(e)$. But g could be itself an exponential cut involving a non-special box. This sequence of non-special boxes must however be finite, because otherwise we would have a cycle that cannot appear in any proof-net (correctness criterion)..

Complexity Bounds. We can now recall the main results on EAL and LAL :

Theorem 1 (Girard [18]) *For every natural number n, there is a polynomial (respectively, elementary function) $e_n : \mathbb{N} \to \mathbb{N}$ such that for every proof-net N of LAL (respectively, EAL) if $N \to^n M$ in the MLBL strategy, then $n \leq e_{\partial(N)}(|N|)$ and $|M| \leq e_{\partial(N)}(|N|)$.*

Recall that binary lists can be represented in LAL with the type: $W = \forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)$. This way, a term of LAL type $t : W \multimap \S^k W$ can be converted to a proof-net, and its application to a list evaluated in polynomial time using $\to_{\mathsf{LAL}}$. However this is still a global evaluation procedure and we want to replace it by optimal reduction.

4 Lamping's Abstract Algorithm

Now we turn to the local reduction procedure. The set of *abstract sharing graphs* Δ_{ASR} is given by the nodes of Figure 6: the 3rd node is called *fan* and is given together with an integer *index* i. A rewriting relation $\to_{\mathsf{ASR}}$ on Δ_{ASR} is defined on Figure 7. Notice that we omit the garbage collection rules. This omission is anyway harmless: the readback procedure (Section 7.2) is not affected by the presence of garbage and the complexity of garbage collection is linear in the size of the graph.

If G is a sharing graph, $fp(G)$ is the set of its *free ports* (dangling edges), while $wp(G)$ is the set of edges which are incident to $\otimes$-nodes. If u is a node of G, then $pp(u)$ is the principal port of u.

To translate proof-nets into sharing graphs we will turn contraction nodes into fans. However we need to choose the indices for the fans. For that, any proof-net N is given together with a *labelling function* $\mathcal{F}$ from the set of its contraction nodes to natural numbers. The translation $\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}$ from proof-nets to abstract sharing graphs will be defined up to such labelling functions. $\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N, \mathcal{F})$ is the graph $G \in \Delta_{\mathsf{ASR}}$ obtained in the following way:

- Replace nodes $R_{\multimap}$ (resp. $L_{\multimap}$) by nodes λ (resp. @),
- Remove boxes and nodes $L_!$, $R_!$, $L_\forall$, $R_\forall$, L_μ, R_μ,
- Replace each contraction node v with a fan-in with index $\mathcal{F}(v)$.

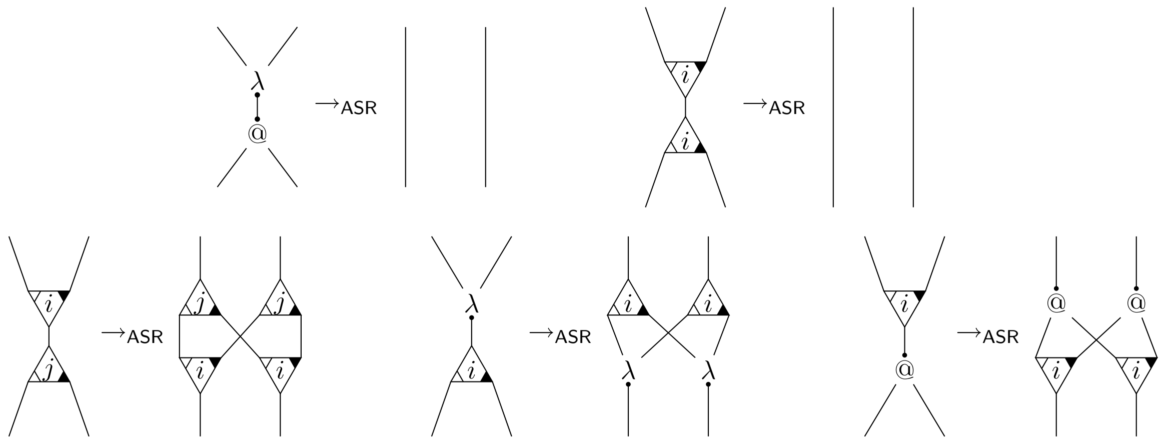


Figure 7. Rewriting rules for sharing graphs.

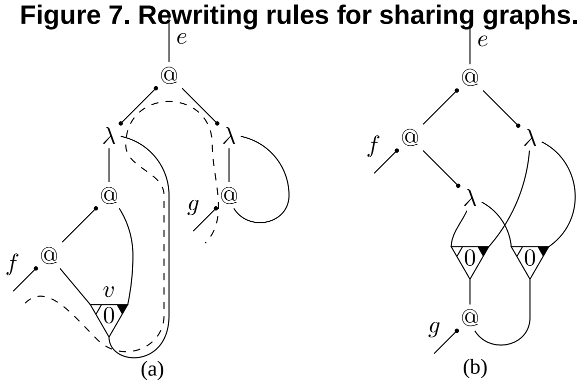


Figure 8. Example.

We denote by $|\mathcal{F}|$ the cardinality of the image of the labelling function $\mathcal{F}$.

We say a labelling function $\mathcal{F}$ for the proof-net N is *compatible with depths* iff $\mathcal{F}(v) = \mathcal{F}(w)$ implies $\partial(v) = \partial(w)$. From now on we will consider only labelling functions which are compatible with depths. Note that in a proof-net reduction step $N \rightarrow_{\mathsf{EAL}} M$, each node of M comes from a unique node of N; a labelling function $\mathcal{F}$ for N then induces in a natural way a labelling function for M, that we will also write $\mathcal{F}$. By the stratification property of EAL, if $\mathcal{F}$ is compatible with depths for N, then so it is for M.

In previous works on light logics and optimal reduction, two particular translations of proof-nets have been considered:

- The *level translation*, LT: the labelling function $\mathcal{F}$ is the one defined by the depths, that is $\mathcal{F}(v) = \partial(v)$.
- The *distinct labelling translation*, DLT: the labelling function $\mathcal{F}$ is the discrete one (each contraction node has a different index),

Observe that the second translation offers the advantage of simplicity, since it does not need the information provided by boxes in N. The first translation, on the other hand, has the advantage that it minimizes the number of indices used to annotate fans in the abstract sharing graph. We will show that these two translations, as well as any one based on a labelling compatible with depths, is sound and complete for beta-reduction. For this purpose we will use as tool a specific *context semantics*. We give on Fig.8(a) an abstract sharing graph that will serve as running example. It is obtained as the DLT of a proof-net corresponding to a derivation of $f :!(A \multimap A) \multimap !(A \multimap A) \multimap B, g :!(A \multimap A) \vdash u : B$, where $u = (\lambda x.f\ x\ x)(\lambda z.g\ z)$. In Fig. 8(b) we give its normal form for $\rightarrow_{\mathsf{ASR}}$.

The concepts of principal port, direct path, simple path, etc. can be easily transferred from proof-nets to sharing graphs. The number of maximal paths in a cut-free sharing graph is bounded:

Lemma 2 *Let G be a cut-free sharing graph and let e be one of its free ports. Then there are at most $|G| + 1$ maximal direct* *paths in the form* $e = e_1, \dots, e_n$.

Proof. Consider any such maximal direct path $e = e_1, \dots, e_n$ and the corresponding sequence of nodes $v_1, \dots, v_{n-1}$. Since there are no cuts in G, there cannot be any e_i which is principal for both v_{i-1} and v_i. This implies $e_1, \dots, e_n$ has a very constrained structure: there is $1 \le j \le n$ such that:

- e_i is principal for v_i whenever $1 \le i < j$.
- e_j is not principal for v_{j-1} (if $j > 1$) nor for v_j (if $j < n$).
- e_i is principal for v_{i-1} whenever $j < i \le n$

So, each such path can be divided into three parts. Now, the third part of the path, namely $e_{j+1}, \dots, e_n$, is completely determined by the first two parts, namely $e_1, \dots, e_j$. But since e_1 is always equal to e, there are at most $|G| + 1$ paths in this form, because every binary tree with n nodes has at most $n + 1$ leaves. This concludes the proof. □

Now we want to bound the complexity of this rewriting procedure and show that it is sound and complete.

5 Context Semantics

5.1 Interpretation of Proof-Nets and Sharing Graphs

Context semantics will be the tool for showing soundness of sharing graph reduction (following [20]). A *context* can be seen as a token carrying a piece of information and travelling around the net [17]. As we are considering a more constrained setting than [20, 19, 16] the contexts can be presented as tuples, as in [8]. This reflects the stratified structure of EAL proof-nets.

Definition 4 (Elementary contexts) *An* elementary context *C of length k is a tuple of stacks $S_1, \dots, S_k, T$ over the alphabet $\{\mathsf{p}, \mathsf{q}\}$. Stacks S_i are called* exponential stacks*, stack T is called* multiplicative stack*. ε denotes the empty stack, xS denotes the stack obtained by pushing x on S and ST denotes the concatenation of stacks S and T. The partial order $\sqsubseteq$ on stacks is the prefix order. We also denote by $\sqsubseteq$ the pointwise order on the product of stacks. Finally $\sqsubseteq_m$ will denote the order on elementary contexts defined by identity on the exponential stacks S_i ($1 \le i \le k$) and $\sqsubseteq$ on the multiplicative stack T.*

Definition 5 (Valid paths) *Let N be in Δ_{EAL} and $\mathcal{F}$ a labelling function, with $k = |\mathcal{F}|$.*

- *A* context *of N relative to $\mathcal{F}$ is a pair (p, C) where p is an edge of N and C is an elementary context of length $k + 1$.*
- *The binary relation $\sim$ on contexts is defined by symmetrically closing the rules in Table 1 and adding for the other (binary) nodes the rule acting as identity on the elementary context (no rule for the W node).*
- *A direct path $e_1, \dots, e_n$ in N is* valid *with respect to two elementary contexts C_1 and C_n iff the nodes along the path transform the context (e_1, C_1) into (e_n, C_n). More precisely, there must be elementary contexts $C_2, \dots, C_{n-1}$ such that $(e_i, C_i) \sim (e_{i+1}, C_{i+1})$ whenever $1 \le i < n$. Then we write $(e_1, C_1) \triangleright (e_n, C_n)$ and say the path $e_1, \dots, e_n$ is* persistent.

Definition 6 (Context semantics) *Given a proof-net N of EAL and a labelling function $\mathcal{F}$, the context semantics $[\![N]\!]_{\mathcal{F}}$ of N is the set of pairs $((e, C), (f, D))$ such that e and f are conclusion edges of N and $(e, C) \triangleright (f, D)$.*

To simplify the notation we will sometimes omit the $\mathcal{F}$ and write $[\![N]\!]$ instead of $[\![N]\!]_{\mathcal{F}}$. Notice that as the transitions are deterministic (see Table 1) and as when reaching a conclusion no transition is possible anymore, if $(e, C) \triangleright (f, D)$ and $(e, C) \triangleright (g, E)$ are both in $[\![N]\!]_{\mathcal{F}}$ then $f = g$ and $D = E$. Therefore the context semantics of N can be seen as a (partial) function on contexts. Notice, however, that there can be two essentially different reasons why the context semantics is undefined on (e, C):

- There could be finitely many valid paths starting in e which are all valid with respect to C and some context D, but none of them ends in a conclusion.
- There are arbitrary long valid paths starting in e which are all valid with respect to C and some context D; this means there is not any context (g, E) such that g is a conclusion edge and $(e, C) \triangleright (g, E)$.

However we will see in Section 5.3 that this second possibility is guaranteed never to occur for proof-nets.

Given a sharing graph G and a partition $\mathcal{F}$ of its fan nodes, its contexts and context semantics $[\![G]\!]$ are defined similarly to that of proof-nets (Table 2). It is then clear that the context semantics is preserved by the translation from proof-nets to sharing graphs:

Table 1. Context Semantics for Proof-nets

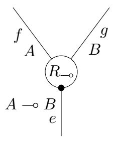
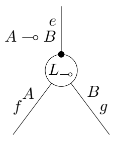
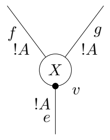
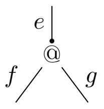
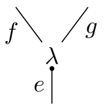
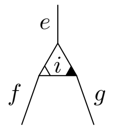
	$(e,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},\mathsf{p}T)) \sim (f,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},T))$ $(e,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},\mathsf{q}T)) \sim (g,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},T))$
	$(e,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},\mathsf{p}T)) \sim (f,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},T))$ $(e,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},\mathsf{q}T)) \sim (g,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},T))$
	$(e,(S_1,\ldots,S_{\mathcal{F}(v)-1},\mathsf{p}S_{\mathcal{F}(v)},S_{\mathcal{F}(v)+1},\ldots,S_{\lvert\mathcal{F}\rvert},T)) \sim (f,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},T))$ $(e,(S_1,\ldots,S_{\mathcal{F}(v)-1},\mathsf{q}S_{\mathcal{F}(v)},S_{\mathcal{F}(v)+1},\ldots,S_{\lvert\mathcal{F}\rvert},T)) \sim (g,(S_1,\ldots,S_{\lvert\mathcal{F}\rvert},T))$

Table 2. Context Semantics for Sharing Graphs

	$(e,(S_1,\ldots,S_n,\mathsf{p}T)) \sim (f,(S_1,\ldots,S_n,T))$ $(e,(S_1,\ldots,S_n,\mathsf{q}T)) \sim (g,(S_1,\ldots,S_n,T))$
	$(e,(S_1,\ldots,S_n,\mathsf{p}T)) \sim (f,(S_1,\ldots,S_n,T))$ $(e,(S_1,\ldots,S_n,\mathsf{q}T)) \sim (g,(S_1,\ldots,S_n,T))$
	$(e,(S_1,\ldots,S_{i-1},\mathsf{p}S_i,S_{i+1},\ldots,S_n,T)) \sim (f,(S_1,\ldots,S_n,T))$ $(e,(S_1,\ldots,S_{i-1},\mathsf{q}S_i,S_{i+1},\ldots,S_n,T)) \sim (g,(S_1,\ldots,S_n,T))$

Proposition 1 *Let N be an* EAL *proof-net and $\mathcal{F}$ a partition of its contraction nodes, then $[\![N]\!]_{\mathcal{F}} = [\![\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N,\mathcal{F})]\!]$.*

We give some examples of contexts in the context semantics of the sharing graph from Fig.8(a):

$$\begin{array}{rclrcl}
(f,\varepsilon,\mathsf{pq}) & \rhd & (g,\mathsf{p},\mathsf{q}); & (f,\varepsilon,\mathsf{qpq}) & \rhd & (g,\mathsf{q},\mathsf{q});\\
(e,\varepsilon,\varepsilon) & \rhd & (f,\varepsilon,\mathsf{qq}); & (g,\mathsf{p},\mathsf{p}) & \rhd & (f,\varepsilon,\mathsf{pp});\\
(g,\mathsf{q},\mathsf{p}) & \rhd & (f,\varepsilon,\mathsf{qpp}). & & &
\end{array}$$

The path corresponding to the first of these contexts is represented on Fig.8(a) by a dashed line.

If P is a set of contexts, P^- denotes the subset of P including only minimal elements (with respect to $\sqsubseteq$). When traversing any node in a sharing graph G, only one particular stack of the underlying context can be modified. Two nodes u and v have the same sort (formally, $ty(u) = ty(v)$) iff they can modify the same stack. For instance @ and λ nodes have the same sort. Given a node u, $ep(u)$ is the set of contexts whose stack corresponding to u is ε.

Lemma 3 (Monotonicity) *Suppose $e_1,\ldots,e_n$ is a direct path valid with respect to $C_1,\ldots,C_k,T$ and $D_1,\ldots,D_k,S$. Moreover, suppose that $E_1,\ldots,E_k,U$ are stacks. Then $e_1,\ldots,e_n$ is valid with respect to $C_1E_1,\ldots,C_kE_k,TU$ and $D_1E_1,\ldots,D_kE_k,SU$.*

Proof. By induction on n. □

Proposition 2 (Minimality) *For every persistent path $e_1, \dots, e_n$ there are elementary contexts C and D such that whenever $e_1, \dots, e_n$ is valid with respect to E and F, $C \sqsubseteq E$ and $D \sqsubseteq F$.*

Proof. By induction on n. □

5.2 Reduction and Context Semantics

Now we consider the behaviour of the context semantics with respect to the reduction of proof-nets and sharing graphs. Let us start with the latter case, which is easier.

Take a look at the rewriting rules for sharing graph. If we focus on the edges involved, we can observe that:

- The annihilation rewriting steps erase one edge, namely the cut. The other four edges involved have *residuals* which are defined in the usual way. The edges which are not directly involved in the rewriting have trivially defined residuals. No edge is created.
- The copying rewriting steps erases one edge but creates another four edges, which are called the *edges created in the rewriting step*. The cut in the redex has no residual.

Let G be a sharing graph and let E be a subset of the edges of G. The direct path $e_1, \dots, e_n$ in G is said to be *long enough for E* iff $e_1, e_n \notin E$.

Lemma 4 (Preservation of Long-Enough Paths) *Suppose G is a sharing graph, $G \rightarrow_{\mathsf{ASR}} H$ by firing a cut e. Then:*

- *If a direct path $e_1, \dots, e_n$ in G is long enough for $\{e\}$ and valid for C and D, then there is a direct path $g_1, \dots, g_m$ in H valid for C and D such that g_1 is the residual of e_1 and g_m is the residual of e_n.*
- *If a direct path $g_1, \dots, g_m$ in H is long enough for the set of edges created in the rewriting step and valid for C and D, then there is a direct path $e_1, \dots, e_n$ in G valid for C and D such that g_1 is the residual of e_1 and g_m is the residual of e_n*

Proof. Consider the rules of Figure 7 and observe that in each case of rewriting step the context semantics partial function of the subgraph concerned is unchanged. □

Proposition 3 *Let G be a sharing graph and $G \rightarrow_{\mathsf{ASR}} H$ then $[\![G]\!] = [\![H]\!]$.*

Proof. Just observe that any conclusion-to-conclusion valid path in G is long enough for any cut, while any conclusion-to-conclusion valid path in H is long enough for the set of edges created in the rewriting step. The thesis follows easily from Lemma 4. □

As to proof-nets the situation is more delicate. It is well-known that geometry of interaction or context semantics are usually not preserved by general proof-net reduction [19, 16, 26]. To deal with this problem we define a partial order $\preccurlyeq$ on context functions. Context semantics will be preserved up to $\succcurlyeq$ but that will be sufficient to obtain a soundness result with respect to lambda-calculus.

Definition 7 *Let f, g be two partial functions on contexts. Then $f \preccurlyeq g$ iff for any context p, C we have:*

1. *If $f(p, C)$ is defined, then so is $g(p, C)$, and $g(p, C) = f(p, C)$,*
2. *If $f(p, C)$ is undefined then either:*
   i. *$g(p, C)$ is undefined,*
   ii. *or $f(p, D)$ is undefined whenever $D \sqsupseteq_m C$.*

The point in subcase 2.ii is that $f(p, C)$ is undefined, but it is not merely because of a lack of information in the multiplicative stack, since no increase of information on this stack can trigger an answer. The behaviour of f on such input is in fact irrelevant for the read-back process that we will define, so the definition of $\preccurlyeq$ does not require anything on $g(p, C)$ in this case.

Lemma 5 *The relation $\preccurlyeq$ is a partial order.*

Proof. The non-obvious fact is whether this relation is transitive. Assume we have $f \preccurlyeq g$ and $g \preccurlyeq h$. Given a context (p, C) and a stack T_0 we will denote by $(p, C) :: T_0$ the context obtained from p, C by replacing the multiplicative stack T of C by TT_0.

Take a context (p, C): if $f(p, C)$ is defined, then so are $g(p, C)$ and $h(p, C)$, and we have $h(p, C) = f(p, C)$. Otherwise if $f(p, C)$ is undefined we have 2 subcases to consider. First, if for any stack T_0, $f((p, C) :: T_0)$ is undefined, then the condition is fulfilled. Otherwise there exists a T_0 such that $f((p, C) :: T_0)$ is defined, and $g(p, C)$ is undefined; then $g((p, C) :: T_0)$ is defined. As $g \preccurlyeq h$ we deduce that $h(p, C)$ is undefined and the condition is fulfilled. Therefore $f \preccurlyeq h$. □

Now we can state the property of context semantics w.r.t. proof-net reduction:

Proposition 4 *Let N be an EAL proof-net and $N \to_{\mathsf{EAL}} M$ then $[\![N]\!] \succcurlyeq [\![M]\!]$.*

Proof. Consider one step of reduction $N \to_{\mathsf{EAL}} M$. We want to define a map ϕ sending each edge e of M to an edge of N of same type. First, every conclusion e of M is naturally associated to a conclusion e' of N, and we define $\phi(e) = e'$. For the other edges we have to distinguish among the different cases of reduction steps of Figure 3; we only describe the map on the edges involved in the reduction step, for the other edges it is defined as expected. Let us consider the various steps (using the notations of Figure 3):

- $\multimap$ reduction step: the edge of M of type A (resp. B) is mapped to the A edge of N incident to the $R_{\multimap}$ node (resp. the B edge of N incident to the $L_{\multimap}$ node).
- Box-box reduction step: the B edge inside the box of M is mapped to the B edge of N incident to the $R_!$ node; the other edges are mapped in the natural way.
- Contraction step: for each X node in M created in the reduction step, the three incident edges with type $!A_i$ are mapped to the $!A_i$ edge of N incident to $L_!$; each edge in the two boxes of M is mapped to the corresponding edge in the box of N; the $!B$ edge of the left (resp. right) box is mapped to the left (resp. right) non-principal edge of X in N.
- μ reduction step: the $A[\mu\alpha.A/\alpha]$ edge of M is mapped to the $A[\mu\alpha.A/\alpha]$ edge of N incident to R_μ.
- $\forall$ reduction step: as in the μ reduction step.

We now define a map from contexts of M to contexts of N, sending a context (p', C') to a context (p, C) of N, with $p = \phi(p')$, and that we also denote as ϕ. If the reduction step considered is any step but the contraction step, then ϕ is simply the identity. In the contraction case: denote by v the contraction node in N involved in this step, by b the box in N to be duplicated, and by b_1, b_2 its two copies in M. Let $i = \mathcal{F}(v)$. Take a context (p', C') in M: if p' is not in one the b_j boxes nor one of their premises, then $\phi(p', C') = (\phi(p'), C')$. If p' is in b_1 (resp. b_2), or one of its $!A_i$ premises, then $\phi(p', C') = (\phi(p'), C)$, where C is obtained from C' by replacing the ith stack S_i by $\mathsf{pS_i}$ (resp. $\mathsf{qS_i}$).

Let us denote by $\sim^*$ the transitive and reflexive closure of the $\sim$ relation in N.

Lemma 6 *Let $N \to_{\mathsf{EAL}} M$. If $(e, C) \sim (f, D)$ is a transition of M, then $\phi(e, C) \sim^* \phi(f, D)$ is obtained by a (possibly empty) sequence of transitions in N.*

Proof. One can check it by examining for each case of reduction step in Figure 3 the various possible transitions in M. Let us just examine here one case for the example.

Consider a contraction reduction step, denote v the contraction node involved in N, and take a transition $(e, C) \sim (f, D)$ in M corresponding to a node v' inside one of the two boxes, say the left one b_1. Assume for instance v' is a contraction node (the other cases are easier). Denote $i = \mathcal{F}(v)$ and $j = \mathcal{F}(v')$. We have $\partial(v') \geq \partial(v) + 1$, therefore as $\mathcal{F}$ is compatible with depths we get $i \neq j$. Then by definition of ϕ: $\phi(e)$ and $\phi(f)$ are incident to a contraction node v'' in N. Moreover $\mathcal{F}(v'') = j$. Therefore $\phi(e, C)$ (resp. $\phi(f, D)$) has same j-th stack as (e, C) (resp. (f, D)) (only the i-th stack has been modified) and it follows that $\phi(e, C) \sim \phi(f, D)$ is a transition of N. □

Consider a valid path in M and a corresponding sequence of contexts $s = (p_1, C_1), \ldots, (p_n, C_n)$ following the transitions of this proof-net. By using Lemma 6 for each of its transitions and concatenating together the paths obtained in N, one obtains a path in N which is direct, and valid because it transforms context $\phi(p_1, C_1)$ into context $\phi(p_n, C_n)$.

It follows that if the function $[\![M]\!]$ is defined on a context (p_0, C_0), then so is $[\![N]\!]$ and we have $[\![M]\!](p_0, C_0) = [\![N]\!](p_0, C_0)$.

Now, assume $[\![M]\!](p_0, C_0)$ is undefined. Let $(p_0, C_0), \ldots, (p_n, C_n)$ be the corresponding sequence of contexts in M, with (p_n, C_n) not admitting any further transition and such that p_n is not a conclusion. As just said there is a valid path in N with sequence of contexts containing (as subsequence) $\phi(p_0, C_0), \ldots, \phi(p_n, C_n)$. If we are in the case of a non-contraction reduction step, then as ϕ acts as the identity on elementary contexts we have $\phi(p_n, C_n) = (\phi(p_n), C_n)$. As in M the context (p_n, C_n) does not admit any further transition, it is the same for $\phi(p_n, C_n)$ in N. Moreover $\phi(p_n)$ is not a conclusion, hence $[\![N]\!](p_0, C_0)$ is undefined. Therefore in this case we have $[\![M]\!] \preccurlyeq [\![N]\!]$.

In the case where the reduction step is a contraction one we keep the same notations for the involved boxes and contraction node defined before. Let us consider again the sequence in M $(p_1, C_1), \ldots, (p_n, C_n)$. As in M the transition on (p_n, C_n) is not defined, this context is entering the principal port of a node v (see Table 1). We have two cases:

- If v is a $R_{\multimap}$ or $L_{\multimap}$ node, then this means that the multiplicative stack T of C_n is ε, so by definition the multiplicative stack of $\phi(p_n, C_n)$ is also empty and thus in N no transition is possible for $\phi(p_n, C_n)$. Therefore in this case $[\![N]\!](p_0, C_0)$ is undefined.
- If v is an X node (contraction), let $k = \mathcal{F}(v)$. Then as the transition is undefined, the kth stack S_k of C_n is ε. Consider D such that $C_1 \sqsubseteq_m D$. Let T be the multiplicative stack of C_1. Then there exists T_0 such that D's multiplicative stack is TT_0. Let us now denote by $(p, C :: T_0)$ the context obtained from (p, C) by replacing the multiplicative stack T of C by TT_0. Then $(p_1, C_1 :: T_0) = (p_1, D)$. The following sequence is obtained by consecutive transitions in M: $(p_1, C_1 :: T_0), \ldots, (p_n, C_n :: T_0)$. Moreover $(p_n, C_n :: T_0)$ has an empty kth stack; hence just as (p_n, C_n), the context $(p_n, C_n :: T_0)$ has no possible transition in N. It follows that $[\![M]\!](p_0, C_0 :: T_0)$ is undefined. Therefore we are in the case 2(ii) of Definition 7.

So we can conclude that $[\![M]\!] \preccurlyeq [\![N]\!]$. □

5.3 Acyclicity

We now describe properties of valid paths in the proof-nets and sharing graphs we are dealing with.

Proposition 5 (Finiteness of valid paths for proof-nets) *Let N be an EAL proof-net. Then there exists an integer k such that for any valid path $e_1, \ldots, e_n$ we have $n \leq k$.*

Proof. This is proved in [8] for $\mathsf{EAL}_\forall$, and the proof can be easily adapted to EAL using the fact that EAL is strongly normalising. □

A *cycle* is a direct path $e_1, \ldots, e_n$ such that:

- $e_1 = e_n$;
- $n \geq 2$;
- $e_1, \ldots, e_n$ is valid with respect to $C = C_0, \ldots, C_k, T$ and $D = D_0, \ldots, D_k, S$;
- For every $0 \leq i \leq k$, either $C_i \sqsubseteq D_i$ or $D_i \sqsubseteq C_i$;
- Either $T \sqsubseteq S$ or $S \sqsubseteq T$.

Proposition 6 (Acyclicity of Proof-Nets) *If N is a proof-net, then its context semantics does not contain any cycle.*

Proof. Indeed if the proof-net N contained a cycle, then by repeatedly composing it with itself one would get valid paths of arbitrary length, which would contradict Proposition 5. □

Proposition 7 *Let N be an EAL proof-net, $G = \mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N, \mathcal{F})$ and $G \to^*_{\mathsf{ASR}} H$. Then there exists an integer k such that: for any valid path $e_1, \ldots, e_n$ of H we have $n \leq k$.*

Proof. First, the statement holds for the paths of G itself because of Prop. 5 and of the fact that any valid path of G can be lifted back to a valid path of N of same length or longer. Then consider H obtained from G by one step of $\to_{\mathsf{ASR}}$. Using Lemma 4 one can show that if H has valid paths of arbitrary length, then so has G, which yields a contradiction. Hence the property is valid for any H such that $G \to_{\mathsf{ASR}} H$. □

6 Complexity

We study the complexity of sharing graph reduction by defining a weight W_G for any sharing graph G. The underlying idea is the following: the weight of G is the sum of the individual weight of each $u \in V_G$, the latter being the number of possible copies of u that are produced during normalisation. We will relate the weight to the number of reduction steps of the sharing graph, and then, for sharing graphs coming from EAL(and LAL), bound the weight by using the properties of proof-nets.

Formally, the weight of an edge u will be defined as the number of different valid paths $e_1, \ldots, e_n$ satisfying certain additional constraints. First of all, e_1 must be $pp(u)$. Secondly, e_n must be:

$$
\begin{array}{rcl}
B_u & = & \big\{C\ |\exists v, D.\big((pp(u), C) \triangleright (pp(v), D)\big) \wedge (C \in ep(u)) \wedge (D \in ep(v)) \wedge (ty(u) = ty(v))\big\} \\
P_u & = & \big\{C\ |\exists q, D.\big((pp(u), C) \triangleright (q, D)\big) \wedge (C \in ep(u)) \wedge (q \in fp(G))\big\} \\
E_u & = & \big\{C\ |\exists q, D.\big((pp(u), C) \triangleright (q, D)\big) \wedge (C \in ep(u)) \wedge (q \in wp(G))\big\}
\end{array}
$$

Figure 9. Definition of B_u, P_u and E_u.

- Either the principal edge of a node v such that $ty(u) = ty(v)$.
- Or an edge in $fp(G) \cup wp(G)$.

This way the weight of u will be exactly the number of copies of u that will eventually appear during reduction of G. This can be characterized by context semantics exploiting Proposition 2:

Definition 8 (Weight) *Let $G \in \Delta_{\mathsf{ASR}}$. Then:*

- *If u is a node of G, then B_u, P_u and E_u are sets of elementary contexts defined in Figure 9.*
- *The weight W_G of G is defined as follows:*

$$W_G = \sum_{u \in V_G} (|B_u^-| + |P_u^-| + |E_u^-| - 1).$$

Notice that W_G can be either a natural number or ω.

Notice how 1 is subtracted from the sum $|B_u^-| + |P_u^-| + |E_u^-|$ when defining W_G. This way, W_G always decreases at any copying normalisation step, as we will see. The weight of a cut-free sharing graph obtained by reducing another sharing graph coming from a proof-net is always null:

Lemma 7 *If $N \in \Delta_{\mathsf{EAL}}$ and $\mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}(N, \mathcal{F}) \rightarrow^*_{\mathsf{ASR}} G$ where $G \in \Delta_{\mathsf{ASR}}$ is a cut-free graph, then $W_G = 0$.*

Proof. Consider any $u \in V_G$ and any direct path starting from u. This path is always simple, since we assume G to be cut-free. Moreover, by Proposition 7, we cannot go on forever building up the path. As a consequence, we will end up at an edge in $fp(G) \cup wp(G)$. This, in particular, implies that $|P_u^-| + |E_u^-| = 1$, while $|B_u^-| = 0$. The thesis follows easily. □

Lemma 8 *If $N \in \Delta_{\mathsf{EAL}}$ is a cut-free proof-net and $\mathcal{F}$ is any partition of its contraction nodes, then $W_{\mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}(N, \mathcal{F})} = 0$.*

Proof. Trivial, since $\mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}(N, \mathcal{F})$ is cut-free whenever N is cut-free and, as a consequence, we can apply Lemma 7. □

Proposition 8 *If $N \in \Delta_{\mathsf{EAL}}$, $G = \mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}(N, \mathcal{F})$, W_G is finite and $G \rightarrow^n_{\mathsf{ASR}} H$, then $n \leq W_G + |G|/2$ and $|H| \leq W_G + |G|$.*

Proof. It is sufficient to observe that:

- Annihilation rewriting steps leave W_G unchanged, while $|G|$ decreases by 2.
- Copying rewriting steps make W_G decrease by 2, while $|G|$ increases by 2.

This implies, by Lemma 7 that W_G is finite and that the total number of copying rewriting steps is $W_G/2$. As a consequence, the size of H is at most $W_G + |G|$. Moreover, the total number of annihilation rewriting steps is $(W_G + |G|)/2$. This completes the proof. □

Given a proof-net $N \in \Delta_{\mathsf{EAL}}$ such that $N \rightarrow_{\mathsf{EAL}} M$, we can study the difference $W_{\mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}(N,\mathcal{F})} - W_{\mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}(M,\mathcal{F})}$. In particular, in the case of the MLBL strategy, the difference can be tightly bounded, because the number of paths that we "lose" at each reduction step can be properly bounded by an appropriate function (with the same order of magnitude as the one from Theorem 1) applied to $|N|$. This implies the weight of the underlying sharing graph cannot decrease too much during MLBL proof-net reduction and, by Lemma 8 and Theorem 1, we get:

Proposition 9 *For every natural number n, there is an elementary function $e_n : \mathbb{N} \rightarrow \mathbb{N}$ such that for every proof-net $N \in \Delta_{\mathsf{EAL}}$, $W_{\mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}(N)} \leq e_{\partial(N)}(|N|)$.*

Proof. First of all, we know that for every natural number n, there are elementary functions $f_n, g_n : \mathbb{N} \rightarrow \mathbb{N}$ such that for every proof-net $N \in \Delta_{\mathsf{EAL}}$ if $N \rightarrow^m_{\mathsf{EAL}} M$, then $m \leq f_{\partial(N)}(|N|)$ and $|M| \leq g_{\partial(N)}(|N|)$. We can build up e_n by induction on n:

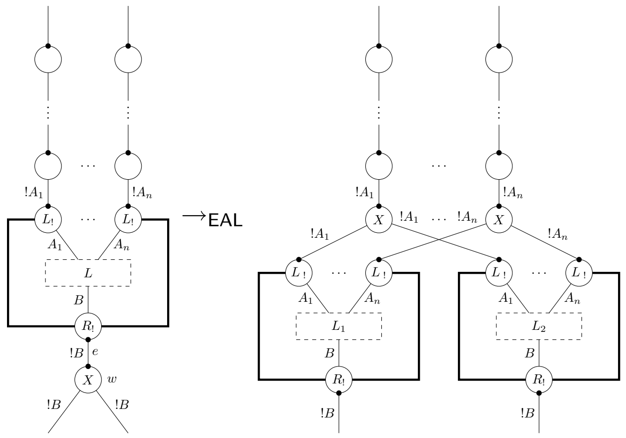


Figure 10.

- $e_0(x) = 0$ for every $x \in \mathbb{N}$. Indeed, let $\partial(N) = 0$. If $N \to_{\mathsf{LAL}} M$ in the modified level-by-level strategy, then $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} = W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(M)}$ and, moreover, $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} = 0$ whenever N is cut-free (by Lemma 8).
- $e_n(x) = f_n(x) \cdot (e_{n-1}(g_n(x)) + 2g_n(x))$ for every $n \geq 1$. Indeed, let $\partial(N) = n \geq 1$. If $N \to_{\mathsf{EAL}} M$ in the MLBL strategy, then $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} - W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(M)} \leq e_{n-1}(|N|) + 2|N|$. This because:
  - At any normalisation step other than copying, $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} = W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(M)}$, as we already pointed out.
  - In the case of copying, we are in the situation depicted in Figure 10. $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} - W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(M)}$ can be bounded as follows:
    - Consider any $u \in V_L$ and any persistent path in N starting from u. Any such path can be mimicked by at least one of the two copies u_1 and u_2 of u appearing in M. In particular, if the path stays inside L, than it can be mimicked by two paths starting in u_1 and u_2, while if the path exits from L, it can be mimicked by exactly one path starting in either u_1 or u_2. By definition of $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(M)}$, the contribution to the weight of these nodes decreases by at most $|N|$.
    - Consider the node $w \in V_N$. Paths starting in w cannot be mimicked by any in V_M. We claim, however, that there cannot be more than $p+1$ such paths, where p is the size of the normal form of $\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(L)$. Indeed, all such paths can be seen as maximal, persistent paths in L. By Proposition 8. the size of the normal form of $\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(L)$ cannot be more than $e_{n-1}(|N|) + |N|$.

  As a consequence, since $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} = 0$ whenever N is cut-free (again by Lemma 8), we can iterate over the inequality $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} - W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(M)} \leq e_{n-1}(|N|) + 2|N|$ obtaining $W_{\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)} \leq f_n(|N|)(e_{n-1}(g_n(|N|)) + 2g_n(|N|))$ (since $g_n(|N|)$ is a bound on the size of any reduct of N).

This concludes the proof. □

Proposition 10 *For every natural number n, there is a polynomial $e_n : \mathbb{N} \to \mathbb{N}$ such that for every proof-net $N \in \Delta_{\mathsf{LAL}}$, $W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)} \leq e_{\partial(N)}(|N|)$.*

Proof. The proof is similar to the one for Proposition 9. First of all, we know that for every natural number n, there are polynomials $f_n, g_n : \mathbb{N} \to \mathbb{N}$ such that for every proof-net $N \in \Delta_{\mathsf{EAL}}$ if $N \to^m_{\mathsf{EAL}} M$, then $m \leq f_{\partial(N)}(|N|)$ and $|M| \leq g_{\partial(N)}(|N|)$. We can build up e_n by induction on n:

- $e_0(x) = 0$ for every $x \in \mathbb{N}$. Indeed, let $\partial(N) = 0$. If $N \to_{\mathsf{EAL}} M$ in the modified level-by-level strategy, then $W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)} = W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(M)}$ and, moreover, $W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)} = 0$ whenever N is cut-free (by Lemma 8).
- $e_n(x) = f_n(x) \cdot (e_{n-1}(g_n(x)) + 2g_n(x))$ for every $n \geq 1$. Indeed, let $\partial(N) = n \geq 1$. If $N \to_{\mathsf{EAL}} M$ in the modified level-by-level strategy, then $W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)} - W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(M)} \leq e_{n-1}(|N|) + 2|N|$. This because we can proceed as in the case of EAL. As a consequence, since $W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)} = 0$ whenever N is cut-free (again by Lemma 8), we can iterate over the inequality $W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)} - W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(M)} \leq e_{n-1}(|N|) + 2|N|$ obtaining $W_{\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)} \leq f_n(|N|)(e_{n-1}(g_n(|N|)) + g_n(|N|))$ (since $g_n(|N|)$ is a bound on the size of any reduct of N).

This concludes the proof. □

By Propositions 10 and 9, we get:

Theorem 2 *For every natural number n, there is a polynomial (resp. elementary function) $e_n : \mathbb{N} \to \mathbb{N}$ such that for every term t typable in* LAL *(resp.* EAL *), if N is a proof-net corresponding to a type derivation of t, then any reduction of the sharing graph $\mathcal{T}^{\mathsf{LAL}}_{\mathsf{ASR}}(N)$ (resp. $\mathcal{T}^{\mathsf{EAL}}_{\mathsf{ASR}}(N)$) has length bounded by $e_{\partial(N)}(|N|)$.*

As an application, if t can be typed in LAL with type $W \multimap \S^k W$ then there exists a polynomial p such that the application of t to the term representing the list w, reduced using sharing graphs, takes at most $p(|w|)$ steps.

7 Soundness

Suppose we are in the following situation:

$$\begin{tikzcd} t \arrow[r, "*"] \arrow[d, "\mathcal{T}"] & u \\ G \arrow[r, "*"] & H \end{tikzcd}$$

In words, we translated a typable term t to a sharing graph G, then normalised G to H. We now need to define a read-back procedure $\mathcal{R}$ that extracts the normal form u of t from H. We have to design a variant of the readback procedures in the literature, *e.g.* [20, 26], since here we are not handling a generic encoding of terms into proof-nets but an encoding based on type derivations.

The procedure $\mathcal{R}^{\mathsf{ASR}}_{\Lambda}$ is defined on sharing graphs, but does not look at the internal structure of the graph itself; rather, the procedure is defined as a set of *queries* to the underlying context semantics. To prove $\mathcal{R}^{\mathsf{ASR}}_{\Lambda}$ is correct, we will show that whenever $\pi : \Gamma \vdash u : A$ is a cut-free type derivation, $\mathcal{R}^{\mathsf{ASR}}_{\Lambda}$ applied to the proof-net M induced by π returns u. This is enough to prove soundness. Indeed, the context semantics of M is essentially the same as the one of H:

$$\begin{array}{ccc} [\![N]\!] & \succcurlyeq & [\![M]\!] \\ \| & & \\ [\![G]\!] & = & [\![H]\!] \end{array}$$

Observe that we could even apply the readback procedure to G, without reducing G to its normal form H. This, however, would make the read-back process less efficient, as the whole computation would be done by the context semantics.

7.1 Structure of Normal Forms

We here recall that for any λ-term t in normal form, there are $n, m \geq 0$ such that $t = \lambda x_1. \ldots .\lambda x_n.yt_1 \ldots t_m$ where $t_1, \ldots, t_m$ are in normal form. This way, we can partition an arbitrary normal form into head subterms. Formally, *head patterns* are syntactical objects defined by the following productions:

$$H[\cdot] ::= [\cdot] \mid \lambda x_1. \ldots .\lambda x_n.yt_1 \ldots t_{i-1}H[\cdot]t_{i+1} \ldots t_m$$

Given a normal form t, a *head subterm* for t is a pair $(H[\cdot], u)$ such that $H[\cdot]$ is a head pattern and $t = H[u]$. The *initial head subterm* for t is $([\cdot], t)$. For example, head subterms of $t = \lambda x.y(\lambda z.zx)w$ are

$$\begin{aligned} X_1 &= ([\cdot], \lambda x.y(\lambda z.zx)w) \\ X_2 &= (\lambda x.y[\cdot]w, \lambda z.zx) \\ X_3 &= (\lambda x.y(\lambda z.zx)[\cdot], w) \\ X_4 &= (\lambda x.y(\lambda z.z[\cdot])w, x); \end{aligned}$$

and the initial head subterm for t is X_1.

A function Ξ on head subterms can be defined such that $\Xi(H[\cdot], u)$ returns a sequence completely describing the internal structure of u. In particular, if $u = \lambda x_1. \ldots .\lambda x_n.yu_1 \ldots u_m$, then $\Xi(H[\cdot], u)$ includes:

- The natural number n.
- Some binding information about the head occurrence of y in u. More specifically, if the occurrence is free in $H[u]$, then $\Xi(H[\cdot], u)$ includes the variable y itself, while if the occurrence is bound, then $\Xi(H[\cdot], u)$ includes a pair $((J[\cdot], v), k)$ locating the binder for y.

- The head subterms corresponding to $u_1, \ldots, u_m$.

Formally, suppose that $t = H[J[u]]$ where

$$\begin{array}{rcl} J[\cdot] & = & \lambda x_1.\ldots.\lambda x_k.yt_1\ldots t_{i-1}K[\cdot]t_{i+1}\ldots t_m; \\ u & = & \lambda z_1.\ldots.\lambda z_p.x_iu_1\ldots u_q. \end{array}$$

In this case, $\Xi(H[J[\cdot]], u)$ will be the sequence

$$(p, (H[\cdot], J[u])), i-1, (L_1[\cdot], u_1), \ldots, (L_q[\cdot], u_q))$$

where for every $1 \leq i \leq q$

$$L_i[\cdot] = H[J[\lambda z_1.\ldots.\lambda z_p.x_iu_1\ldots u_{i-1}[\cdot]u_{i+1}\ldots u_q]].$$

Now, suppose that $t = H[u]$ with

$$u = \lambda x_1.\ldots.\lambda x_k.yt_1\ldots t_m.$$

and the head occurrence of y in u is free in t. In this case, $\Xi(H[\cdot], u)$ will be the sequence

$$(k, y, (L_1[\cdot], t_1), \ldots, L_m[\cdot], t_m)).$$

where, for every $1 \leq i \leq m$,

$$L_i[\cdot] = H[\lambda x_1.\ldots.\lambda x_p.yt_1l\ldots t_{i-1}[\cdot]t_{i+1}\ldots t_m].$$

As an example,

$$\begin{array}{rcl} \Xi(X_1) & = & (1, y, X_2, X_3) \\ \Xi(X_2) & = & (1, X_2, 0, X_4) \\ \Xi(X_3) & = & (0, w) \\ \Xi(X_4) & = & (0, X_1, 0) \end{array}$$

For every head subterm X, the sequence $\Xi(X)$ is univocally determined. As a consequence, Ξ can be seen as a total function acting on the space of head subterms.

Let us now forget about the internal structure of head subterms and focus on the Ξ function. If we know which is the initial subterm of a given term t, we can reconstruct the whole term t by invoking Ξ many times. For example, $\Xi(X_1)$ tells us that t has 1 abstraction, its head variable is y and the two subterms corresponds to X_2 and X_3. Calling again Ξ on X_2, we can get some information on the first of the two subterm: it has one abstraction, the head variable is bound by the same abstraction and it has another subterm, corresponding to X_4. And so on.

7.2 The Read-back Procedure

The map Ψ is defined as a function on pairs in the form (N, C) (where N is a proof-net and C is a context for N). The value of Ψ on $(N, (p, C_1, \ldots, C_k, T))$ is defined as follows:

- Let n be the least number such that $[\![N]\!]$ is defined on $(p, C_1, \ldots, C_k, T\mathsf{q}^n)$. Suppose $(p, C_1, \ldots, C_k, T\mathsf{q}^n) \triangleright (r, D_1, \ldots, D_k, S)$.
- If $S = R\mathsf{p}\mathsf{q}^l\mathsf{p}\mathsf{q}^m$, then return the value

$$(n, (N, (r, D_1, \ldots, D_k, R\mathsf{p})), l, Q_1, \ldots, Q_m)$$

  where for every $1 \leq i \leq m$

$$Q_i = (N, (r, D_1, \ldots, D_k, R\mathsf{p}\mathsf{q}^l\mathsf{p}\mathsf{q}^i\mathsf{p}))$$

- If $S = \mathsf{q}^m$, then return the value

$$(n, r, Q_1, \ldots, Q_m)$$

  where for every $1 \leq i \leq m$

$$Q_i = (N, (r, D_1, \ldots, D_k, \mathsf{q}^i\mathsf{p}))$$

Observe that in computing Ψ, we use nothing but the context semantics of N. Moreover, we have the following property:

Proposition 11 *If $\pi : \Gamma \vdash t : A$ is a cut-free type derivation, then there is a function Φ_π mapping head subterms of t to pairs in the form (N_π, C) (where N_π is the proof-net induced by π and C is a context for N_π) satisfying the following conditions:*

1. *$\Phi_\pi([\cdot], t) = (N_\pi, (p, \varepsilon, \ldots, \varepsilon))$ where p is the conclusion of N_π.*
2. *For every head subterm X of t, if $\Xi(X) = (i, Y, j, X_1, \ldots, X_n)$, then $\Psi(\Phi_\pi(X)) = (i, \Phi_\pi(Y), j, \Phi_\pi(X_1), \ldots, \Phi_\pi(X_n))$*
3. *For every head subterm X of t, if $\Xi(X) = (i, y, X_1, \ldots, X_n)$, then $\Psi(\Phi_\pi(X)) = (i, p, \Phi_\pi(X_1), \ldots, \Phi_\pi(X_n))$ where p is the edge of N_π corresponding to the variable y.*

Proof. By induction on π:

- Suppose that:
$$\pi : \overline{x : A \vdash x : A}\ A$$
The only head subterm of x is the initial subterm $([\cdot], x)$. Moreover, $\Xi([\cdot], x) = (0, x)$. We defined $\Phi_\pi([\cdot], x) = (N_\pi, (p, \varepsilon, \ldots, \varepsilon))$, where p is the conclusion of N_π. Indeed, if we apply Ψ to $(N_\pi, (p, \varepsilon, \ldots, \varepsilon))$ we obtain $(0, p)$ as a result and, clearly, p is the edge of N_π corresponding to variable x.
- Suppose that:
$$\pi : \frac{\rho : \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\ R_{\multimap}$$
The head subterms of $\lambda x.t$ are:
  - The initial head subterm $([\cdot], \lambda x.t)$.
  - The head subterm $(\lambda x.H[\cdot], u)$ whenever $(H[\cdot], u)$ is a (non-initial) head subterm for t.

  The following equalities hold:
$$\Xi([\cdot], \lambda x.t) = \begin{cases} (n+1, y, (\lambda x.J_1[\cdot], u_1), \ldots, (\lambda x.J_m[\cdot], u_m)) \\ \quad \text{if } \Xi([\cdot], t) = (n, y, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m)) \\ (n+1, ([\cdot], \lambda x.t), l+1, (\lambda x.J_1[\cdot], u_1), \ldots, (\lambda x.J_m[\cdot], u_m)) \\ \quad \text{if } \Xi([\cdot], t) = (n, ([\cdot], \lambda x.t), l, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m)) \\ (n+1, ([\cdot], \lambda x.t), 0, (\lambda x.J_1[\cdot], u_1), \ldots, (\lambda x.J_m[\cdot], u_m)) \\ \quad \text{if } \Xi([\cdot], t) = (n, x, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m)) \end{cases}$$
$$\Xi(\lambda x.H[\cdot], u) = \begin{cases} (n, y, (\lambda x.J_1[\cdot], u_1), \ldots, (\lambda x.J_m[\cdot], u_m)) \\ \quad \text{if } \Xi(H[\cdot], t) = (n, y, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m)) \\ (n, (\lambda x.K[\cdot], v), l, (\lambda x.J_1[\cdot], u_1), \ldots, (\lambda x.J_m[\cdot], u_m)) \\ \quad \text{if } \Xi(H[\cdot], t) = (n, (K[\cdot], v), l, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m)) \\ (n, ([\cdot], \lambda x.t), 0, (\lambda x.J_1[\cdot], u_1), \ldots, (\lambda x.J_m[\cdot], u_m)) \\ \quad \text{if } \Xi(H[\cdot], t) = (n, x, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m)) \end{cases}$$
  Now, let p be the main conclusion of N_ρ, $r, r_1, \ldots, r_h$ be the premises of N_ρ (where r is the premise corresponding to x and $r_1, \ldots, r_h$ are the other ones). Analogously, let s be the main conclusion of N_π and $q_1, \ldots, q_h$ be the premises of N_π. We define Φ_π from Φ_ρ (which exists by IH) as follows:
$$\Phi_\pi([\cdot], \lambda x.t) = (N_\pi, (s, \varepsilon, \ldots, \varepsilon))$$
$$\Phi_\pi(\lambda x.H[\cdot], u) = \begin{cases} (N_\pi, (s, C_1, \ldots, C_k, \mathsf{q}S)) \\ \quad \text{if } \Phi_\rho(H[\cdot], u) = (N_\rho, (p, C_1, \ldots, C_k, S) \\ (N_\pi, (s, C_1, \ldots, C_k, \mathsf{p}S)) \\ \quad \text{if } \Phi_\rho(H[\cdot], u) = (N_\rho, (r, C_1, \ldots, C_k, S) \\ (N_\pi, (q_i, C_1, \ldots, C_k, S)) \\ \quad \text{if } \Phi_\rho(H[\cdot], u) = (N_\rho, (r_i, C_1, \ldots, C_k, S) \end{cases}$$
  We are now able to prove conditions 1 to 3. By definition, it is clear that condition 1 is satisfied. Condition 2: suppose that $\Xi(X) = (i, Y, j, X_1, \ldots, X_n)$. We can distinguish four cases, depending on the shape of X and the way we have defined Ξ:

- The following equalities hold:

$$\begin{aligned}
X &= ([\cdot], \lambda x.t)\\
Y &= ([\cdot], \lambda x.t)\\
\forall 1 \leq a \leq n. X_a &= (\lambda x.J_a[\cdot], u_a)\\
\Xi([\cdot], t) &= (i-1, ([\cdot], t), j-1, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m))
\end{aligned}$$

  By definition, $\Phi_\pi(X) = (N_\pi, (s, \varepsilon, \ldots, \varepsilon))$. Moreover, by IH,

$$\Psi(\Phi_\rho([\cdot], t)) = (i-1, \Phi_\rho([\cdot], t), j-1, \Phi_\rho(J_1[\cdot], u_1), \ldots, \Phi_\rho(J_m[\cdot], u_m))$$

  The computation of $\Psi(\Phi_\pi(X))$ is carried out very similarly to the one of $\Psi(\Phi_\rho([\cdot], t))$. By exploiting the way we have defined Φ_π and the way N_π is built starting from N_ρ, we easily get the desired equality:

$$\Psi(\Phi_\pi(X)) = (i, \Phi_\pi(Y), j, \Phi_\pi(X_1), \ldots, \Phi_\rho(X_n))$$

- The following equalities hold:

$$\begin{aligned}
X &= ([\cdot], \lambda x.t)\\
Y &= ([\cdot], \lambda x.t)\\
j &= 0\\
\forall 1 \leq a \leq n. X_a &= (\lambda x.J_a[\cdot], u_a)\\
\Xi([\cdot], t) &= (i-1, x, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m))
\end{aligned}$$

  By definition, $\Phi_\pi(X) = (N_\pi, (s, \varepsilon, \ldots, \varepsilon))$. Moreover, by IH,

$$\Psi(\Phi_\rho([\cdot], t)) = (i-1, r, \Phi_\rho(J_1[\cdot], u_1), \ldots, \Phi_\rho(J_m[\cdot], u_m))$$

  The computation of $\Psi(\Phi_\pi(X))$ is carried out very similarly to the one of $\Psi(\Phi_\rho([\cdot], t))$. By exploiting the way we have defined Φ_π and the way N_π is built starting from N_ρ, we easily get the desired equality:

$$\Psi(\Phi_\pi(X)) = (i, \Phi_\pi(Y), 0, \Phi_\pi(X_1), \ldots, \Phi_\rho(X_n))$$

- The following equalities hold:

$$\begin{aligned}
X &= (\lambda x.H[\cdot], u)\\
Y &= (\lambda x.K[\cdot], v)\\
\forall 1 \leq a \leq n. X_a &= (\lambda x.J_a[\cdot], u_a)\\
\Xi(H[\cdot], t) &= (i, (K[\cdot], v), j, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m))
\end{aligned}$$

  By IH:

$$\Psi(\Phi_\rho(H[\cdot], t)) = (i, \Phi_\rho(K[\cdot], v), j, \Phi_\rho(J_1[\cdot], u_1), \ldots, \Phi_\rho(J_m[\cdot], u_m))$$

  The computation of $\Psi(\Phi_\pi(X))$ is carried out very similarly to the one of $\Psi(\Phi_\rho(H[\cdot], t))$. By exploiting the way we have defined Φ_π and the way N_π is built starting from N_ρ, we easily get the desired equality:

$$\Psi(\Phi_\pi(X)) = (i, \Phi_\pi(Y), j, \Phi_\pi(X_1), \ldots, \Phi_\rho(X_n))$$

- The following equalities hold:

$$\begin{aligned}
X &= (\lambda x.H[\cdot], u)\\
Y &= ([\cdot], \lambda x.t)\\
j &= 0\\
\forall 1 \leq a \leq n. X_a &= (\lambda x.J_a[\cdot], u_a)\\
\Xi(H[\cdot], t) &= (i, x, (J_1[\cdot], u_1), \ldots, (J_m[\cdot], u_m))
\end{aligned}$$

By IH:

$$\Psi(\Phi_\rho(H[\cdot],t)) = (i, r, \Phi_\rho(J_1[\cdot],u_1), \ldots, \Phi_\rho(J_m[\cdot],u_m))$$

The computation of $\Psi(\Phi_\pi(X))$ is carried out very similarly to the one of $\Psi(\Phi_\rho(H[\cdot],t))$. By exploiting the way we have defined Φ_π and the way N_π is built starting from N_ρ, we easily get the desired equality:

$$\Psi(\Phi_\pi(X)) = (i, \Phi_\pi(Y), j, \Phi_\pi(X_1), \ldots, \Phi_\rho(X_n))$$

Now, suppose that $\Xi(X) = (i, y, X_1, \ldots, X_n)$. We can distinguish two cases, depending on the shape of X and the way we have defined Ξ:

- The following equalities hold:

$$\begin{array}{rcl} X & = & ([\cdot], \lambda x.t) \\ \forall 1 \leq a \leq n.X_a & = & (\lambda x.J_a[\cdot], u_a) \\ \Xi([\cdot],t) & = & (i-1, y, (J_1[\cdot],u_1), \ldots, (J_m[\cdot],u_m)) \end{array}$$

  By definition, $\Phi_\pi(X) = (N_\pi, (s, \varepsilon, \ldots, \varepsilon))$. Moreover, by IH,

$$\Psi(\Phi_\rho([\cdot],t)) = (i-1, r_y, \Phi_\rho(J_1[\cdot],u_1), \ldots, \Phi_\rho(J_m[\cdot],u_m))$$

  The computation of $\Psi(\Phi_\pi(X))$ is carried out very similarly to the one of $\Psi(\Phi_\rho([\cdot],t))$. By exploiting the way we have defined Φ_π and the way N_π is built starting from N_ρ, we easily get the desired equality:

$$\Psi(\Phi_\pi(X)) = (i, q_y, \Phi_\pi(X_1), \ldots, \Phi_\rho(X_n))$$

- The following equalities hold:

$$\begin{array}{rcl} X & = & (\lambda x.H[\cdot], u) \\ \forall 1 \leq a \leq n.X_a & = & (\lambda x.J_a[\cdot], u_a) \\ \Xi(H[\cdot],t) & = & (i, y, (J_1[\cdot],u_1), \ldots, (J_m[\cdot],u_m)) \end{array}$$

  By IH:

$$\Psi(\Phi_\rho(H[\cdot],t)) = (i, r_y \Phi_\rho(J_1[\cdot],u_1), \ldots, \Phi_\rho(J_m[\cdot],u_m))$$

  The computation of $\Psi(\Phi_\pi(X))$ is carried out very similarly to the one of $\Psi(\Phi_\rho(H[\cdot],t))$. By exploiting the way we have defined Φ_π and the way N_π is built starting from N_ρ, we easily get the desired equality:

$$\Psi(\Phi_\pi(X)) = (i, y, \Phi_\pi(X_1), \ldots, \Phi_\rho(X_n))$$

- Suppose that:

$$\pi : \frac{\rho : \Gamma \vdash t : A \quad \sigma : \Delta, x : B \vdash u : C}{\Gamma, \Delta, y : A \multimap B \vdash u\{yt/x\} : C} \; L_{\multimap}$$

  We can proceed as in the previous case.

Ths concludes the proof. □

A readback procedure $\mathcal{R}_\Lambda^{\mathsf{ASR}} : \Delta_{\mathsf{ASR}} \to \Lambda$ is defined by iteratively calling Ψ. This, by Proposition 11, produces the normal form of the term we started from. Moreover, $\mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}$ can be given the status of a total binary relation from Θ_{EAL} to Δ_{ASR}. We finally get:

Theorem 3 (Soundness) *The Θ_{EAL}-graph rewriting system $(\Theta_{\mathsf{EAL}}, \Delta_{\mathsf{ASR}}, \to_{\mathsf{ASR}}, \mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}, \mathcal{R}_\Lambda^{\mathsf{ASR}})$ is sound.*

8 Completeness

Theorem 4 (Completeness) *The Θ_{EAL}-graph rewriting system $(\Theta_{\mathsf{EAL}}, \Delta_{\mathsf{ASR}}, \to_{\mathsf{ASR}}, \mathcal{T}_{\mathsf{ASR}}^{\mathsf{EAL}}, \mathcal{R}_\Lambda^{\mathsf{ASR}})$ is complete.*

Proof. It is sufficient to observe that Theorem 2 implies that reducing G will lead to a normal form H. Then it follows from the soundness result of Section 7 that $\mathcal{R}_\Lambda^{\mathsf{ASR}}(H) = u$. □

9 Conclusions

We proved that Lamping's abstract algorithm is sound and complete for beta reduction of EAL (and LAL) typable pure lambda-terms. Moreover, the number of graph interaction steps is shown to be bounded by the same kind of bounds holding for proof-net normalisation. All these results have been established by exploiting context semantics. In particular, complexity results have been inferred in an innovative way, being inspired by [14].

Further work includes the extension of the approach to general optimal reduction. In the full algorithm, however, relative bounds should take the place of absolute bounds, since any pure lambda term can be reduced.

Acknowledgements. We are grateful to Harry Mairson and Simone Martini for stimulating discussions and advice on optimal reduction. We also wish to thank Vincent Atassi for useful conversations.

References

[1] A. Asperti, P. Coppola, and S. Martini. (Optimal) duplication is not elementary recursive. *Information and Computation*, 193:21–56, 2004.

[2] A. Asperti and H.G. Mairson. Parallel beta reduction is not elementary recursive. *Information and Computation*, 170(1):49–80, 2001.

[3] A. Asperti and L. Roversi. Intuitionistic light affine logic. *ACM Transactions on Computational Logic*, 3(1):1–39, 2002.

[4] Andrea Asperti. Light Affine Logic. In *Proceedings of LICS'98*, pages 300–308. IEEE Computer Society, 1998.

[5] Andrea Asperti and Stefano Guerrini. *The Optimal Implementation of Functional Programming Languages*, volume 45 of *Cambridge Tracts in Theoretical Computer Science*. CUP, 1998.

[6] V. Atassi, P. Baillot, and K. Terui. Verification of Ptime reducibility for system F terms via Dual Light Affine Logic. In *Proceedings of CSL'06*, number 4207 in LNCS, pages 150–166. Springer, 2006.

[7] P. Baillot and K. Terui. A feasible algorithm for typing in Elementary Affine Logic. In *Proceedings of TLCA'05*, volume 3461 of *LNCS*, pages 55–70. Springer, 2005.

[8] Patrick Baillot and Marco Pedicini. Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2):1–31, 2001.

[9] Patrick Baillot and Kazushige Terui. Light types for polynomial time computation in lambda-calculus. In *Proceedings of LICS'04*, pages 266–275, 2004.

[10] Paolo Coppola, Ugo Dal Lago, and Simona Ronchi Della Rocca. Elementary affine logic and the call-by-value lambda calculus. In *Proc. TLCA'05*, volume 3461 of *LNCS*, pages 131–145. Springer, 2005.

[11] Paolo Coppola and Simone Martini. Optimizing optimal reduction. a type inference algorithm for elementary affine logic. *ACM Transactions on Computational Logic*, 7(2):219–260, 2006.

[12] Paolo Coppola and Simona Ronchi Della Rocca. Principal typing for lambda calculus in elementary affine logic. *Fundamenta Informaticae*, 65(1-2):87–112, 2005.

[13] U. Dal Lago and M. Hofmann. Quantitative models and implicit complexity. In *Proceedings of FSTTCS'05*, volume 3821 of *LNCS*, pages 189–200. Springer, 2005.

[14] Ugo Dal Lago. Context semantics, linear logic and computational complexity. Arxiv cs.LO/0510092. Extended abstract appeared in Proceedings IEEE LICS'06, pp.169-178, 2005.

[15] Ugo Dal Lago and Patrick Baillot. Light affine logic, uniform encodings and polynomial time. *Mathematical Structures in Computer Science*, 16(4):713–733, 2006.

[16] V. Danos and L. Regnier. Proof-nets and Hilbert space. In *Advances in Linear Logic*, pages 307–328. CUP, 1995.

[17] Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal lambda-machines. *Theor. Comput. Sci.*, 227(1-2):79–97, 1999.

[18] J.-Y. Girard. Light linear logic. *Information and Computation*, 143:175–204, 1998.

[19] Jean-Yves Girard. Geometry of interaction 1: interpretation of system F. In *Proc. Logic Colloquium '88*, pages 221–260, 1989.

[20] Geaorges Gonthier, Martn Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In *Proceedings of ACM POPL'92*, pages 15–26, 1992.

[21] Stefano Guerrini, Simone Martini, and Andrea Masini. Coherence for sharing proof-nets. *Theoretical Computer Science*, 294(3):379–409, 2003.

[22] Yves Lafont. Interaction combinators. *Inf. Comput.*, 137(1):69–101, 1997.

[23] John Lamping. An algorithm for optimal lambda calculus reduction. In *Proceedings of ACM POPL'90*, pages 16–30, 1990.

[24] Olivier Laurent and Lorenzo Tortora de Falco. Obsessional cliques: a semantic characterization of bounded time complexity. In *Proceedings of LICS'06*, pages 179–188. IEEE Computer Society Press, 2006.

[25] Julia L. Lawall and Harry G. Mairson. Optimality and inefficiency: What isn't a cost model of the lambda calculus? In *Proc. 1996 ACM SIGPLAN ICFP*, pages 92–101, 1996.

[26] Harry Mairson. From Hilbert spaces to Dilbert spaces: Context semantics made simple. In *Proc. 22nd Conference FSTTCS*, volume 2556 of *LNCS*, pages 2–17, 2002.

[27] A. S. Murawski and C.-H. L. Ong. Discreet games, light affine logic and ptime computation. In *Proceedings of CSL'00*, volume 1862 of *LNCS*, pages 427–441. Springer, 2000.

[28] Marco Pedicini and Francesco Quaglia. A parallel implementation for optimal lambda-calculus reduction. In *Proceedings of ACM PPDP'00*, pages 3–14, 2000.